USNO/AA Technical Note 2010-04

Testing Coordinate Frame Transformations NOVAS vs SOFA

Alice Monet, George Kaplan, & William Harris
14 July 2010

The Standards of Fundamental Astronomy (IAU 2009a; SOFA)[1] library is the official collection of approved software for positional astronomy, operating under the auspices of International Astronomical Union (IAU) Division 1 (Fundamental Astronomy). Both Fortran and C libraries are available. An international SOFA Reviewing Board manages the collection.

Generally, the Naval Observatory Vector Astrometry Software (Kaplan et al. 2009; NOVAS)[2] is independent of SOFA although both software libraries include code that is similar to two International Earth Rotation and Reference Systems Service (IERS) Fortran modules.[3] *NU2000A* and *iau2000a* (Fortran and C, respectively), which evaluate the full 1,365-term IAU 2000A nutation series in NOVAS 3.0, are based on the IERS subroutine *NU2000A*. *EECT2000* and *ee_ct*, which evaluate the "complementary terms" in the equation of the equinoxes, are based on IERS function *EECT2000*. The corresponding modules in SOFA are *iau_NUT00A*, *iauNut00a*, *iau_EECT00*, and *iauEect00*.

The document *SOFA Tools for Earth Attitude* (IAU 2009b), also known as the "SOFA Cookbook," contains several Fortran examples of the transformation between terrestrial and celestial coordinate systems. This technical note examines how one of those examples plays out in both NOVAS and SOFA.

Goal

These tests were designed to compare the transformation from the Geocentric Celestial Reference System (GCRS) to International Terrestrial Reference System (ITRS) using the IAU 2000A/2006 models for precession and nutation. For the Fortran test, we compared the Celestial Intermediate Origin (CIO) based method in NOVAS_F3.0g[4] at full accuracy[5] with the "IAU 2006/2000A, CIO based, using classical angles" example in the SOFA Cookbook. For the C test, we used **SOFA_Test.c**, attached to this report. The goal was to verify that the NOVAS libraries, which are (mostly) independent of SOFA, produced results that agree with

[1] http://www.iausofa.org/index.html
[2] http://www.usno navy.mil/USNO/astronomical-applications/software-products/novas
[3] http://tai.bipm.org/iers/conv2003/conv2003_c5 html
[4] beta version "g" of NOVAS F3.0
[5] `CALL ciotio/CALL hiacc` for `mode = 0`

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **14 JUL 2010**	2. REPORT TYPE	3. DATES COVERED **00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE **Testing Coordinate Frame Transformations NOVAS vs SOFA**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Naval Observatory,3450 Massachusetts Avenue, N.W.,Washington,DC,20392**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Standards of Fundamental Astronomy (SOFA)1 library is the official collection of approved software for positional astronomy, operating under the auspices of International Astronomical Union (IAU) Division 1 (Fundamental Astronomy). Both Fortran and C libraries are available. An international SOFA Reviewing Board manages the collection. Generally, the Naval Observatory Vector Astrometry Software2 (NOVAS) is independent of SOFA although both software libraries include code that is similar to two International Earth Rotation and Reference System Service (IERS) Fortran modules.3 NU2000A and iau2000a (Fortran and C, respectively), which evaluate the full 1,365-term IAU 2000A nutation series in NOVAS 3.0, are based on the IERS subroutine NU2000A. EECT2000 and ee_ct, which evaluate the "complementary terms" in the equation of the equinoxes, are based on IERS function EECT2000. The corresponding modules in SOFA are iau_NUT00A, iauNut00a, iau_EECT00, and iauEect00. The document SOFA Tools for Earth Attitude, also known as the "SOFA Cookbook", contains several Fortran examples of the transformation between terrestrial and celestial coordinate systems. This technical note examines how one of those examples plays out in both NOVAS and SOFA.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**26**	

their SOFA counterparts at a level that is at least an order of magnitude better than the best observational results.

Procedure for Fortran Tests

The programs used for the Fortran tests were **TERCEL-TEST.f** and **SOFA_TEST.f**; copies of which are in Addendum I and Addendum II, respectively. The input parameters, which were taken from the SOFA Cookbook, were the following:

- Universal Time: `UT1 = 2400000.5 + 54195.4999991658` days (Julian date)
  The UT1 value is divided into two parts, i.e., two separate arguments, because of the large number of significant digits needed for precise results. The best agreement between NOVAS and SOFA was obtained when UT1 was split in the exactly the same place; splitting the date differently produced differences of about 3 microarcseconds (μas).
- Difference between Terrestrial Time (TT) and UT1: `ΔT = 65.25607389` s
  SOFA does not use ΔT; this value is the difference between the TT and UT1 Julian dates in the SOFA example, expressed in seconds.
- Polar coordinates: `XP = 0.0349282`, `YP = 0.4833163` arcsec
- Celestial Intermediate Pole (CIP) offsets: `DX = 0.1725`, `DY = -0.265` arcsec

The SOFA subroutine *iau_NUT06A* includes small corrections to the nutation series arising from the P03 precession (Capitaine, Wallace, & Chapront 2003; hereafter P03) that are not used in the NOVAS calculations. The corrections amount to only a few microarcseconds for current dates.

NOVAS does not directly produce an overall GCRS-to-ITRS rotation matrix as SOFA does. The NOVAS rotation matrix was constructed simply by passing the three vectors, (1,0,0), (0,1,0), and (0,0,1), in succession through subroutine *TERCEL*.

A series of tests were done, with and without corrections for polar motion, precession and nutation, and the P03 correction in SOFA, and the resulting rotation matrices were compared. The computations were executed on a 32-bit Mac system running the Leopard (OS X 10.5) operating system.

Results of Fortran Tests

Table 1 shows that the latest Fortran releases of NOVAS and SOFA agree at the sub-microarcsecond level in the transformation between the celestial and terrestrial reference systems when the same Earth orientation parameters and conventions are used. In this case, including the P03 corrections in the SOFA nutation adds a discrepancy on the order of 1.4 μas. Inclusion of the CIP offsets and polar motion does not significantly add to the differences in the two formulations, as long as the parameters used are identical in the two cases. Use of the external **CIO_RA** file in the NOVAS calculation adds about 0.05 μas to the difference for the above case, while using equinox mode for the NOVAS computations does not have a significant effect on the results.

Table 1. Comparison of NOVAS F3.0 and SOFA Fortran

Corrections Applied			Other Options		
Polar Motion	CIP Offsets	P03 Terms	External **CIO_RA**	`EQINOX` mode	Difference (μas)
No	No	No	No	No	0.25814
No	No	Yes	No	No	1.6752
No	Yes	Yes	No	No	1.6728
Yes	Yes	Yes	No	No	1.6735
Yes	Yes	No	No	No	0.28679
Yes	Yes	No	Yes	No	0.34369
Yes	Yes	No	No	Yes	0.28644

The results presented in Table 1 were obtained by computing the GCRS to ITRS transformations for the single time discussed in the SOFA Cookbook. Therefore, the values should be typical. Comparisons of future releases should compare transformations for multiple times to estimate the range of possible differences.

Procedure for C Tests

The program used for the C tests was **SOFA_Test.c**; a copy of which is in Addendum III. **SOFA_Test.c** is basically a line-for-line transliteration of the Fortran **SOFA-TEST.f** that uses the SOFA C functions. At the end of the file, a C function *terceltest*, based on the Fortran program **terceltest**, is included. C function *terceltest* was run separately from the bulk of **SOFA_Test.c** using the following input parameters:

- Universal Time: `UT1 = 2400000.5 + 54195.4999991658` days (Julian date)
  The UT1 value is divided into two parts, i.e., two separate arguments, because of the large number of significant digits needed for precise results. The best agreement between NOVAS and SOFA was obtained when UT1 was split in the exactly the same place.

- Difference between TT and UT1: `ΔT = 65.25607389` s
  SOFA does not use ΔT; this value is the difference between the TT and UT1 Julian dates in the SOFA example, expressed in seconds.

- Polar coordinates: `XP = 0.0349282`, `YP = 0.4833163` arcsec

- CIP offsets: `DX = 0.1725, DY = -0.265` arcsec

The SOFA function *iauNut06a* includes small corrections to the nutation series arising from the P03 precession that are not used in the NOVAS calculations. The corrections amount to only a few microarcseconds for current dates.

NOVAS does not directly produce an overall GCRS-to-ITRS rotation matrix as SOFA does. The NOVAS rotation matrix was constructed simply by passing the three vectors, (1,0,0), (0,1,0), and (0,0,1), in succession through function *ter2cel*.

A series of tests were done, with and without corrections for polar motion, precession and nutation, and the P03 correction in SOFA. The output of the C version of **terceltest** and **SOFA_Test** were compared with output from the corresponding Fortran programs. As with the Fortran tests, the C computations were executed on a 32-bit Mac system running the Leopard (OS X 10.5) operating system.

Results of the C Tests

The outputs of the SOFA C tests were identical with those produced by the Fortran versions. Addendum III includes a sample output from the C tests.

References

Capitaine, N., Wallace, P. T., Chapront, J. 2003, A&A, 412, 567 (P03)

Kaplan, G., Bangert, J., Bartlett, J., Puatua, W., & Monet, A. 2009, *User's Guide to NOVAS 3.0*, USNO Circular 180[6] (Washington, DC: USNO) (NOVAS) http://www.usno.navy.mil/USNO/astronomical-applications/software-products/novas

IAU. 2009a, Standards of Fundamental Astronomy, (SOFA) http://www.iausofa.org/

IAU. 2009b, *SOFA Tools for Earth Attitude*, Software version 4, Document revision 1.1 (SOFA Cookbook) http://www.iausofa.org/sofa_pn.pdf

IERS. 2003, Conventions 2003: Chapter 5 Transformation Between the Celestial and Terrestrial Systems (Frankfurt, Germany: BKG) http://tai.bipm.org/iers/conv2003/conv2003_c5.html

[6] http://www.usno navy.mil/USNO/astronomical-applications/publications/circ-180

Addendum I: NOVAS Fortran Test Code

The NOVAS Fortran code for the comparison of NOVAS and SOFA is contained in `PROGRAM terceltest` below. This program reads the contents of **tercel-test-input.dat**, which is also listed below. The output of `terceltest` may be directed to a file, such as **tercel_matrix**, which can then be read by **SOFA-TEST.f.**

Program File: TERCEL-TEST.f

```
      PROGRAM terceltest

C
C
C---PURPOSE:    Transform vectors from ITRS to GCRS for comparing NOVAS
C               with SOFA
C
C---REFERENCES: None
C
C---INPUT
C   ARGUMENTS:  num   spacer read from file
C                     (INTEGER)
C               tjdh  TDB or TT Julian Date, higher part, read from file
C                     (DOUBLE PRECISION)
C               tjdl  TDB or TT Julian Date, lower part, read from file
C                     (DOUBLE PRECISION)
C               xp    conventionally-defined x coordinate of CIP
C                     with respect to ITRS pole, arcseconds, read from
C                     file
C                     (DOUBLE PRECISION)
C               yp    conventionally-defined y coordinate of CIP
C                     with respect to ITRS pole, arcseconds, read from
C                     file
C                     (DOUBLE PRECISION)
C               vec1  position vector, geocentric equatorial rectangular
C                     coordinates, referred to ITRS axes, read from file
C                     (DOUBLE PRECISION)
C
C---OUTPUT
C   ARGUMENTS:  vec2  position vector, geocentric equatorial rectangular
C                     coordinates, referred to GCRS axes
C                     (DOUBLE PRECISION)
C---COMMON
C   BLOCKS:     None
C
C---ROUTINE
C   CALLED:     SUBROUTINE setdt    (NOVAS)
C               SUBROUTINE celpol   (NOVAS)
C               SUBROUTINE ciotio   (NOVAS)
C               SUBROUTINE eqinox   (NOVAS)
C               SUBROUTINE hiacc    (NOVAS)
C               SUBROUTINE tercel   (NOVAS)
C
C---COMPILING:  compile with NOVAS, solsys2, jpljubs
C               have JPLEPH in working directory
```

```
C                have CIO_RA.TXT in working directory for tests w/
C                external CIO_RA file
C
C---VER./DATE/
C   PROGRAMMER: v1.0/02-09/AM  (USNO/AA) initial version
C                v1.5/09-10/JLB (USNO/AA) additional documentation
C
C---NOTES       1. Input agruments read from tercel-test-input.dat.
C                2. For best agreement with SOFA, split the date at
C                   the same point, i.e. tjdh = 2400000.5
C                3. Send output to file, tercel_matrix, to be read by
C                   SOFA_TEST.f, which does comparison.
C                4. Initial test variations
C                   A. no polar motion, no CIP offsets, no P03
C                      corrections (SOFA), no CIO_RA, cio mode
C                   B. no polar motion, no CIP offsets, P03
C                      corrections (SOFA), no CIO_RA, cio mode
C                   C. no polar motion, CIP offsets, P03
C                      corrections (SOFA), no CIO_RA, cio mode
C                   D. polar motion, CIP offsets, P03
C                      corrections (SOFA), no CIO_RA, cio mode
C                   E. polar motion, CIP offsets, no P03
C                      corrections (SOFA), no CIO_RA, cio mode
C                   F. polar motion, CIP offsets, no P03
C                      corrections (SOFA), CIO_RA, cio mode
C                   G. polar motion, CIP offsets, no P03
C                      corrections (SOFA), no CIO_RA, equinox mode
C-----------------------------------------------------------------------

      DOUBLE PRECISION tjdh, tjdl, xp, yp, delt, vec1(3),
     . vec2(3), tjd, dx, dy
      INTEGER num, mode

      DATA delt /65.25607389d0/, num / 0 /
      CALL setdt (delt)

C-----dx, dy are celestial pole offsets to be used
C-----use 1st pair to invoke correction
C-----use 2nd pair to ignore correction

      dx = +0.1750d0
      dy = -0.2259d0
C     dy = 0d0
C     dx = 0d0

C-----Open the input file of Julian dates,CIO coords,ITRS vector

      OPEN (UNIT = 15, FILE = 'tercel-test-input.dat',
     . STATUS = 'OLD', ACCESS = 'SEQUENTIAL')

   10 READ (15,*, END = 20) num, tjdh, tjdl, xp, yp, vec1

C-----Set transformation method, accuracy level, and UT1-UTC.
C-----CIO-based method (ciotio) is used for most tests
C-----Equinox-base method (eqinox) is used in one case
```

```
      tjd = tjdh + tjdl

      CALL celpol (tjd,2,dx,dy)
C     CALL ciotio
      CALL eqinox
      CALL hiacc

C-----Rotate vec1 from ITRS to GCRS = vec2

      CALL TERCEL(tjdh,tjdl,xp,yp,vec1,vec2)
      WRITE (*,101) num, vec2
         GO TO 10
   20 CONTINUE

  101 FORMAT (i1,3(4x,f20.17))

      STOP
      END
```

Input Data File: tercel-test-input.dat

```
1 2400000.5   54195.49999916581146 +0.0349282d0 +0.4833163d0 1d0  0d0  0d0
1 2400000.5   54195.49999916581146 +0.0349282d0 +0.4833163d0 0d0  1d0  0d0
1 2400000.5   54195.49999916581146 +0.0349282d0 +0.4833163d0 0d0  0d0  1d0
```

Addendum II: SOFA Fortran Test Code

The SOFA Fortran code for the comparison of NOVAS and SOFA is contained in **SOFA_TEST.f**, which is based on Example 5.5 “IAU 2006/2000A, CIO based, using classical angles” in the SOFA Cookbook; a copy of the program file follows. The program reads two input data files: **tercel_matrix** and **sofa_matrix**, which are listed below. At the end of this section, a sample of its output is provided.

Program File: SOFA-TEST.f

```
      PROGRAM SOFA_Test

*
*
*---PURPOSE:   Compare performance of NOVAS F3.0 & SOFA Fortran
*              libraries
*
*---REFERENCES: IAU. 2009, SOFA Tools for Earth Attitude, Software
*                   version 4, Document revision 1.1 (SOFA Cookbook)
*                   http://www.iausofa.org/sofa_pn.pdf
*
*---INPUT
*   ARGUMENTS:  TERMAT  =  3x3 matrix read from tercel_matrix
*                          (DOUBLE PRECISION)
*               SOFMAT  =  3x3 matrix read from sofa_matrix
*                          (DOUBLE PRECISION)
*---OUTPUT
*   ARGUMENTS:  NONE
*
*---COMMON
*   BLOCKS:     NONE
*
*---ROUTINES
*   CALLED:     DOUBLE PRECISION FUNCTION iau_ANP (SOFA)
*               SUBROUTINE iau_BPN2XY  (SOFA)
*               SUBROUTINE iau_C2IXYS  (SOFA)
*               SUBROUTINE iau_C2IXYS  (SOFA)
*               SUBROUTINE iau_CAL2JD  (SOFA)
*               SUBROUTINE iau_CP  (SOFA)
*               SUBROUTINE iau_CR  (SOFA)
*               SUBROUTINE iau_D2TF  (SOFA)
*               SUBROUTINE iau_DAT  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_ERA00  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAD03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAE03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAF03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAJU03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAL03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FALP03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAMA03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAME03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAOM03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FAPA03  (SOFA)
*               DOUBLE PRECISION FUNCTION iau_FASA03  (SOFA)
```

```
*                DOUBLE PRECISION FUNCTION iau_FAUR03  (SOFA)
*                DOUBLE PRECISION FUNCTION iau_FAVE03  (SOFA)
*                SUBROUTINE iau_FW2M  (SOFA)
*                SUBROUTINE iau_IR  (SOFA)
*                SUBROUTINE iau_NUT00A  (SOFA)
*                SUBROUTINE iau_NUT06A  (SOFA)
*                DOUBLE PRECISION FUNCTION iau_OBL06  (SOFA)
*                SUBROUTINE iau_PFW06  (SOFA)
*                SUBROUTINE iau_PM  (SOFA)
*                SUBROUTINE iau_PNM06A  (SOFA)
*                SUBROUTINE iau_POM00  (SOFA)
*                SUBROUTINE iau_RM2V  (SOFA)
*                SUBROUTINE iau_RX  (SOFA)
*                SUBROUTINE iau_RXR  (SOFA)
*                SUBROUTINE iau_RY  (SOFA)
*                SUBROUTINE iau_RZ  (SOFA)
*                DOUBLE PRECISION FUNCTION iau_S06  (SOFA)
*                DOUBLE PRECISION FUNCTION iau_SP00  (SOFA)
*                SUBROUTINE iau_TR  (SOFA)
*                SUBROUTINE iau_XYS06A  (SOFA)
*                SUBROUTINE iau_ZR  (SOFA)
*                SUBROUTINE REPMAT (internal)
*                DOUBLE PRECISION FUNCTION DROT (internal)
*                SUBROUTINE TRANSPOSE (internal)
*
*---COMPILING:  SOFA_Test.f must be compiled with appropriate SOFA
*                  modules. SOFA modules are available as discrete
*                  files. Downloading the entire library and
*                  concatenating it into a single file may be easiest.
*
*---VER./DATE/
*   PROGRAMMER: v1.0/02-09/AM  (USNO/AA) initial version
*                v1.1/09-10/JLB (USNO/AA) additional documentation
*
*---NOTES       1. Input agruments read from tercel_matrix, sofa_matrix
*                2. SOFA_Test does not call all of the routines listed
*                   above directly; some are called by routines called by
*                   SOFA_Test. A fuller than usual list is given here to
*                   ensure the user has all the necessary SOFA files.
*------------------------------------------------------------------------

*  Preliminaries from SOFA Cookbook, section 5.1

*  SOFA examples

      IMPLICIT NONE

*  Arcseconds to radians
      DOUBLE PRECISION AS2R
      PARAMETER ( AS2R = 4.848136811095359935899141D-6 )

*  2Pi
      DOUBLE PRECISION D2PI
      PARAMETER ( D2PI = 6.283185307179586476925287D0 )
```

```
*  PM, R1(3,3) through R5(3,3), TERMAT (3,3), TERTRANS(3,3),
*  SOFMAT, SOFTRANS, DROT, DJMJDATE, UTHI, UTLO
*  are not from Cookbook

      CHARACTER PM
      INTEGER IY, IM, ID, IH, MIN, J
      INTEGER IHMSF(4)
      DOUBLE PRECISION SEC, XP, YP, DUT1,
     :                 DDP80, DDE80, DX00, DY00, DX06, DY06,
     :                 DJMJD0, DATE, TIME, UTC, DAT,
     :                 TAI, TT, TUT, UT1, RP(3,3), DP80, DE80,
     :                 DPSI, DEPS, EPSA, RN(3,3), RNPB(3,3),
     :                 EE, GST, RC2TI(3,3), RPOM(3,3),
     :                 RC2IT(3,3), X, Y, S,
     :                 RC2I(3,3), ERA, DP00, DE00, RB(3,3),
     :                 RPB(3,3), V1(3), V2(3), DDP00, DDE00
      DOUBLE PRECISION R1(3,3), R2(3,3), R3(3,3), R4(3,3), R5(3,3)
      DOUBLE PRECISION iau_OBL80, iau_EQEQ94, iau_ANP, iau_GMST82,
     :                 iau_ERA00, iau_SP00, iau_EE00, iau_GMST00,
     :                 iau_S06,TERMAT(3,3), TERTRANS(3,3),
     :                 SOFMAT(3,3), SOFTRANS(3,3)
      DOUBLE PRECISION DROT, DJMJDATE, UTHI, UTLO

*  Open and read file containing output from tercel.f
C.....
      OPEN (UNIT = 16, FILE = 'tercel_matrix', STATUS = 'OLD',
     . ACCESS = 'SEQUENTIAL')
      READ (16,90) TERMAT
   90 FORMAT ( 1x, 3e24.12 )
      WRITE (*,'( 3(f20.15) )') TERMAT

*  Open and read file containing sofa_matrix
C
      OPEN (UNIT = 17, FILE = 'sofa_matrix', STATUS = 'OLD',
     . ACCESS = 'SEQUENTIAL')
      READ (17,95) SOFMAT
   95 FORMAT ( 1x, 3f24.17 )
C     WRITE (*,'( 3(f20.15) )') SOFMAT
C.....


*  Initalize variables using values from SOFA Cookbook Preliminaries

*  UTC.
      IY = 2007
      IM = 4
      ID = 5
      IH = 12
      MIN = 0
      SEC = 0D0
      WRITE ( *, '(1X,''date'',I6.4,2(''/'',I2.2))' ) IY, IM, ID
      WRITE ( *, '(1X,''time'',I4,I3,F5.1,''  UTC'')' ) IH, MIN, SEC

*  Polar motion (arcsec->radians).
      XP = 0.0349282D0 * AS2R
```

```
      YP = 0.4833163D0 * AS2R
      WRITE ( *, '(/1X,''X_p ='',SP,F13.9,'' arcsec'')' ) XP/AS2R
      WRITE ( *, '( 1X,''Y_p ='',SP,F13.9,'' arcsec'')' ) YP/AS2R

*  UT1-UTC (s).
      DUT1 = -0.072073685D0
      WRITE ( *, '(/1X,''UT1-UTC ='',SP,F16.12,'' s'')' ) DUT1

*  Nutation corrections wrt IAU 1976/1980 (mas->radians).
      DDP80 = -55.0655D0 * AS2R/1000D0
      DDE80 =  -6.3580D0 * AS2R/1000D0
      WRITE ( *, '(/1X,''dDpsi (1980) ='',SP,F13.9,'' arcsec'')' )
     :                                                       DDP80 / AS2R
      WRITE ( *, '( 1X,''dDeps (1980) ='',SP,F13.9,'' arcsec'')' )
     :                                                       DDE80 / AS2R

*  CIP offsets wrt IAU 2000A (mas->radians).
      DX00 =  0.1725D0 * AS2R/1000D0
      DY00 = -0.2650D0 * AS2R/1000D0
      WRITE ( *, '(/1X,''dX (2000) ='',SP,F13.9,'' arcsec'')' )
     :                                                        DX00 / AS2R
      WRITE ( *, '( 1X,''dY (2000) ='',SP,F13.9,'' arcsec'')' )
     :                                                        DY00 / AS2R

*  CIP offsets wrt IAU 2006/2000A (mas->radians).
*  First set from SOFA Cookbook, 2nd set to ignore correction
      DX06 = 0.1750D0 * AS2R/1000D0
      DY06 = -0.2259D0 * AS2R/1000D0
C     DX06 = 0d0
C     DY06 = 0d0  * AS2R/1000D0
      WRITE ( *, '(/1X,''dX (2006) ='',SP,F13.9,'' arcsec'')' )
     :                                                        DX06 / AS2R
      WRITE ( *, '( 1X,''dY (2006) ='',SP,F13.9,'' arcsec'')' )
     :                                                        DY06 / AS2R

*  TT (MJD).
      CALL iau_CAL2JD ( IY, IM, ID, DJMJD0, DATE, J )
      TIME = ( 60D0*(60D0*DBLE(IH) + DBLE(MIN)) + SEC ) / 86400D0
      UTC = DATE + TIME
      CALL iau_DAT ( IY, IM, ID, TIME, DAT, J )
      TAI = UTC + DAT/86400D0
      TT = TAI + 32.184D0/86400D0
      WRITE ( *, '(/1X,''TT  = 2400000.5 +'',F22.15 )' ) TT

*  UT1.
      TUT = TIME + DUT1/86400D0
      UT1 = DATE + TUT
      WRITE ( *, '(/1X,''UT1 = 2400000.5 +'',F22.15 )' ) UT1


*  Following SOFA Cookbook, IAU 2006/2000A, CIO based,
*  classical angles, section 5.5 with additional intermediate output


*  =========================
```

```
*  IAU 2006/2000A, CIO-based
*  ========================

      WRITE ( *, '(/1X,''   ========================''' //
     :            '/1X,''4) IAU 2006/2000A, CIO-based''' //
     :            '/1X,''   ========================'')' )

*  CIP and CIO, IAU 2006/2000A.

C-----Report inputs to CIP calculation


      CALL iau_XYS06A ( DJMJD0, TT, X, Y, S )

*  Add CIP corrections.
      X = X + DX06
      Y = Y + DY06

*  Report CIP and s.
      WRITE ( *, '(/1X,''X ='',SP,F22.15)' ) X/AS2R
      WRITE ( *, '( 1X,''Y ='',SP,F22.15)' ) Y/AS2R
      WRITE ( *, '( 1X,''s ='',SP,F13.9,'' arcsec'')' ) S/AS2R

*  GCRS to CIRS matrix.
      CALL iau_C2IXYS ( X, Y, S, RC2I )

C     WRITE (*,*) 'X,Y,S/AS2R: ',X, Y, S/AS2R

*  Report.
      CALL REPMAT ( 'NPB matrix, CIO-based (RC2I)', RC2I )

*  Earth rotation angle.
C
C----- Set TUT to the value used in Tercel

      WRITE (*,*) 'DJMJD0, DATE, TUT: ', DJMJD0, DATE, TUT

C       DJMJDATE = 2400000.5d0
C       TUT = 54195.4999991658d0
C     UTHI = 2454195.0d0
C       UTLO = 0.99999916581146d0
C
      ERA = iau_ERA00 ( DJMJD0, DATE+TUT )

C     ERA = iau_ERA00 ( UTHI, UTLO )
C     ERA = iau_ERA00 ( DJMJD0+DATE, TUT )

      WRITE ( *, '(1X)' )
      WRITE ( *, '(1X,''ERA ='',F19.16,'' rad'' )' ) ERA
      WRITE ( *, '(1X,''    ='',F16.12,'' deg'' )' ) ERA*360D0/D2PI
      CALL iau_D2TF ( 9, ERA/D2PI, PM, IHMSF )
      WRITE ( *, '(1X,''    ='',3I3.2,''.'',I9.9 )' ) IHMSF

*  Form celestial-terrestrial matrix (no polar motion yet).
      CALL iau_CR ( RC2I, RC2TI )
```

```
      CALL iau_RZ ( ERA, RC2TI )

*  Report.
      CALL REPMAT ( 'celestial to terrestrial matrix (no polar motion)',
     :                                                           RC2TI )
      CALL iau_CR (RC2TI,R3)

*  Polar motion matrix (TIRS->ITRS, IERS 2003).
      CALL iau_POM00 ( XP, YP, iau_SP00(DJMJD0,TT), RPOM )

*  Form celestial-terrestrial matrix (including polar motion).
      CALL iau_RXR ( RPOM, RC2TI, RC2IT )


*  Compare results

C Transpose TERMAT
      CALL TRANSPOSE ( TERMAT, TERTRANS )
      CALL TRANSPOSE ( SOFMAT, SOFTRANS )

*  Report.
      CALL REPMAT ( 'celestial to terrestrial matrix', RC2IT )
      CALL REPMAT ( 'tercel_transposed matrix', TERTRANS )

C     WRITE ( *, '(1X)' )
C     WRITE (*,'( 3(f20.15) )') TERMAT

*  Copy for later comparison.
      CALL iau_CR ( RC2IT, R4 )

*  Compare result to TERCEL_Transposed matrix
      WRITE ( *, '(1X)' )
      WRITE (*,'( 1x,''w/ pm result vs tercel ='',e20.10,'' uas'')')
     . DROT ( R4, TERTRANS ) *1D6 / AS2R

* Compare SOFA w/P03 and SOFA w/o P03
      WRITE ( *, '(1X)' )
      WRITE (*,'( 1x,''sofa w/p03 vs w/o p03 ='',e20.10,'' uas'')')
     . DROT ( R4, SOFTRANS ) *1D6 / AS2R

      END

*----------------------------------------------------------------------*

      SUBROUTINE REPMAT ( S, R )

*
*
*---PURPOSE:     Format and print 3x3 matrix
*
*---REFERENCES:  None
*
*---INPUT
*   ARGUMENTS:   S = description of matrix
*                    (CHARACTER string)
```

```
*                 R = 3x3 matrix
*                     (DOUBLE PRECISION array)
*
*---OUTPUT
*   ARGUMENTS:   None
*
*---COMMON
*   BLOCKS:      None
*
*---ROUTINES
*   CALLED:      None
*
*
*---VER./DATE/
*   PROGRAMMER: v1.0/02-09/AM  (USNO/AA) initial version
*               v1.1/09-10/JLB (USNO/AA) additional documentation
*
*-----------------------------------------------------------------------

      IMPLICIT NONE
      CHARACTER*(*) S
      DOUBLE PRECISION R(3,3)
      WRITE (*,'(/1X,A/SP,2(3F22.17/),3F22.17)') S,R(1,1),R(1,2),R(1,3),
     :                                             R(2,1),R(2,2),R(2,3),
     :                                             R(3,1),R(3,2),R(3,3)
C     WRITE ( *, '(1X)' )
C     WRITE (*,*) S
C     WRITE (*,'(3f22.17)') R
      END

*----------------------------------------------------------------------*

      DOUBLE PRECISION FUNCTION DROT ( RMA, RMB )

*
*
*---PURPOSE:     Express the difference between two rotation matrices
*                RMA, RMB as an amount of rotation R about some
*                arbitrary axis
*
*---REFERENCES:  None
*
*---INPUT
*   ARGUMENTS:   RMA = First 3x3 matrix for comparison
*                      (DOUBLE PRECISION array)
*                RMA = Second 3x3 matrix for comparison
*                      (DOUBLE PRECISION array)
*
*---OUTPUT
*   ARGUMENTS:   DROT = amount of rotation between RMA, RMB
*                       (DOUBLE PRECISION)
*
*---COMMON
*   BLOCKS:      None
*
```

```
*---ROUTINES
*   CALLED:      iau_TR   (SOFA)
*                iau_RXR  (SOFA)
*                iau_RM2V (SOFA)
*                iau_PM   (SOFA)
*
*---VER./DATE/
*   PROGRAMMER: v1.0/02-09/AM  (USNO/AA) initial version
*               v1.1/09-10/JLB (USNO/AA) additional documentation
*
*-----------------------------------------------------------------------

      IMPLICIT NONE
      DOUBLE PRECISION RMA(3,3), RMB(3,3)

      DOUBLE PRECISION RMBT(3,3), RM(3,3), RV(3), R

*  Multiply the first matrix by the inverse of the second.
      CALL iau_TR ( RMB, RMBT )
      CALL iau_RXR ( RMBT, RMA, RM )

*  Express the result as an r-vector.
      CALL iau_RM2V ( RM, RV )

*  Return the magnitude (the amount of rotation in radians).
      CALL iau_PM ( RV, R )
      DROT = R

      END

*----------------------------------------------------------------------*
      SUBROUTINE TRANSPOSE (R, RT)

*
*
*---PURPOSE:     Transpose 3x3 matrix R ---> RT
*
*---REFERENCES:  None
*
*---INPUT
*   ARGUMENTS:   R = 3x3 matrix to be transposed
*                    (DOUBLE PRECISION array)
*
*---OUTPUT
*   ARGUMENTS:   RT = 3x3 matrix, transpose of R
*                     (DOUBLE PRECISION array)
*
*---COMMON
*   BLOCKS:      None
*
*---ROUTINES
*   CALLED:      None
*
*---VER./DATE/
*   PROGRAMMER: v1.0/02-09/AM  (USNO/AA) initial version
```

```
*                v1.1/09-10/JLB (USNO/AA) additional documentation
*
*-----------------------------------------------------------------------


      IMPLICIT NONE
      DOUBLE PRECISION  R(3,3), RT(3,3)
      INTEGER i, j
*
      DO i = 1, 3
           DO j = 1, 3
             RT(i,j) = R(j,i)
           ENDDO
      ENDDO
*
      END
```

Input Data File: tercel_matrix

```
1     0.97310431770109063     0.23036382622411070    -0.00070316348296544
1    -0.23036380044102267     0.97310457063635536     0.00011854535854669
1     0.00071156016155651     0.00004662641201635     0.99999974575402495
```

Input Data File: sofa_matrix

```
1    +0.97310431757363003    +0.23036382624733387    -0.00070333224579995
1    -0.23036379880468646    +0.97310457073561185    +0.00012088727913663
1    +0.00071226387930021    +0.00004438635469672    +0.99999974535497649
```

Output Data

```
   0.973104317701091   0.230363826224111  -0.000703163482965
  -0.230363800441023   0.973104570636355   0.000118545358547
   0.000711560161557   0.000046626412016   0.999999745754025
 date  2007/04/05
 time  12  0  0.0  UTC

 X_p = +0.034928200 arcsec
 Y_p = +0.483316300 arcsec

 UT1-UTC = -0.072073685000 s

 dDpsi (1980) = -0.055065500 arcsec
 dDeps (1980) = -0.006358000 arcsec

 dX (2000) = +0.000172500 arcsec
 dY (2000) = -0.000265000 arcsec

 dX (2006) = +0.000175000 arcsec
 dY (2006) = -0.000225900 arcsec

 TT  = 2400000.5 + 54195.500754444445192

 UT1 = 2400000.5 + 54195.499999165811460

    =========================
```

```
 4) IAU 2006/2000A, CIO-based
    ========================

 X =  +146.915146447359746
 Y =    +9.155115079288397
 s = -0.002200475 arcsec

 NPB matrix, CIO-based (RC2I)
  +0.99999974633944499  -0.00000000513882246  -0.00071226473007242
  -0.00000002647522726  +0.99999999901497483  -0.00004438524282713
  +0.00071226472959891  +0.00004438525042571  +0.99999974535441982
 DJMJD0, DATE, TUT:    2400000.5000000000        54195.000000000000
0.49999916581383103

 ERA = 0.2324515536471452 rad
     = 13.318492965240 deg
     = 00 53 16.438311658

 celestial to terrestrial matrix (no polar motion)
  +0.97310431757657001  +0.23036382623316559  -0.00070333281884719
  -0.23036379878963870  +0.97310457073901657  +0.00012088854957536
  +0.00071226472959891  +0.00004438525042571  +0.99999974535441982

 celestial to terrestrial matrix
  +0.97310431770097838  +0.23036382622458479  -0.00070316348220013
  -0.23036380044149363  +0.97310457063624356  +0.00011854536661437
  +0.00071156016266793  +0.00004662640399541  +0.99999974575402439

 tercel_transposed matrix
  +0.97310431770109063  +0.23036382622411070  -0.00070316348296544
  -0.23036380044102267  +0.97310457063635536  +0.00011854535854669
  +0.00071156016155651  +0.00004662641201635  +0.99999974575402495

 w/ pm result vs tercel =    0.1673896075E+01 uas

 sofa w/p03 vs w/o p03 =    0.4843068713E+06 uas
```

Addendum III: C Version of SOFA Test Code

The C version of the SOFA test code consists of **SOFA_Test.c** and its accompanying header file, **SOFA_Test.h**. The SOFA Cookbook does not include examples in C. Therefore, **SOFA_Test.c** is basically a line-for-line transliteration of the Fortran **SOFA-TEST.f** that uses the SOFA C functions. At the end of the file, a C function *terceltest*, based on the Fortran program **terceltest**, is included.

SOFA_Test reads two input files: **tercel-matrix.txt** and **sofa-matrix.txt**. It produced the output reproduced at the end of this addendum.

terceltest reads a single input file: **tercel-test-input.dat**.

Program Header File: SOFA_Test.h

```
#include "stdio.h"
#include "sofa.h"
#include "novas.h"

void SOFA_Test ();
void repmat (char *s,double r[3][3]);
double drot (double rma[3][3],double rmb[3][3]);
void transpose (double r[3][3],double rt[3][3]);
void terceltest ();
```

Program File: SOFA_Test.c

```
#include "SOFA_Test.h"


void SOFA_Test ()
{

/*  Arcseconds to radians*/
    double as2r = 4.848136811095359935899141e-6;

/*  2Pi */
   double twopi = 6.283185307179586476925287;

   char pm;
   short year, month, day, hour, minute;
   int IHMSF[4];
   short i,j;

   double jd, seconds, xp, yp, dut1, ddp80, dde80, dx00, dy00, dx06, dy06,
      djmjd0, date, time, utc, dat, tai, TT, tut, ut1, result1,result2,
      rc2ti[3][3], rpom[3][3],rc2it[3][3], x, y, s, rc2i[3][3], era;
   double r3[3][3], r4[3][3];
   double termat[3][3], tertrans[3][3],sofmat[3][3], softrans[3][3];

   FILE *tercel_data,*sofa_data;

   tercel_data = fopen ("tercel_matrix.txt","r");
```

```
   for (i=0; i<3; i++)
      for (j=0; j<3; j++)
         fscanf (tercel_data,"%lf", &termat[j][i]);
   repmat ("termat", termat);

   sofa_data = fopen ("sofa_matrix.txt","r");
   for (i=0; i<3; i++)
      for (j=0; j<3; j++)
         fscanf (sofa_data,"%lf", &sofmat[j][i]);
   repmat ("sofmat", sofmat);

/*  utc. */
   year = 2007;
   month = 4;
   day = 5;
   hour = 12;
   minute = 0;
   seconds = 0.0;

   printf ("date  %4i %2.2i  %2.2i\n", year, month, day);
   printf ("time  %4i %2.2i  %5.1f UTC\n", hour, minute, seconds);

/*  Polar motion (arcsec->radians). */
   xp = 0.0349282 * as2r;
   yp = 0.4833163 * as2r;
   printf ("X_p =  %13.9f arcsec\n",xp/as2r);
   printf ("Y_p =  %13.9f arcsec\n",yp/as2r);

/*  ut1-utc (s). */
   dut1 = -0.072073685;
   printf ("ut1-utc = %16.12f\n", dut1);

/*  Nutation corrections wrt IAU 1976/1980 (mas->radians). */
   ddp80 = -55.0655 * as2r / 1000.0;
   dde80 =  -6.3580 * as2r / 1000.0;

   printf ("dDpsi (1980) = %13.9f arcsec\n",ddp80 / as2r);
   printf ("dDeps (1980) = %13.9f arcsec\n",dde80 / as2r);

/*  CIP offsets wrt IAU 2000A (mas->radians). */
   dx00 =  0.1725 * as2r / 1000.0;
   dy00 = -0.2650 * as2r / 1000.0;
   printf ("dx (2000) = %13.9f arcsec\n",dx00 / as2r);
   printf ("dy (2000) = %13.9f arcsec\n",dy00 / as2r);

/*  CIP offsets wrt IAU 2006/2000A (mas->radians). */
   dx06 =  0.1750 * as2r / 1000.0;
   dy06 = -0.2259 * as2r / 1000.0;

   printf ("dx (2006) = %13.9f arcsec\n",dx06 / as2r);
   printf ("dx (2006) = %13.9f arcsec\n",dx06 / as2r);

/*  TT (MJD). */
   iauCal2jd (year, month, day, &djmjd0, &date);
```

```
   time = (60.0 * ((60.0 * (double) hour) + (double) minute) + seconds) /
86400.0;
   utc = date + time;
   iauDat (year, month, day, time, &dat);
   tai = utc + dat / 86400.0;
   TT = tai + 32.184 / 86400.0;
   printf ("TT  = 2400000.5 + %22.15f\n", TT);

/*  ut1. */
   tut = time + dut1 / 86400.0;
   ut1 = date + tut;
   printf ("UT1  = 2400000.5 + %22.15f\n", ut1);

/*  =========================
*  IAU 2006/2000A, CIO-based
*  ========================= */

   printf ("=========================\n");
   printf ("IAU 2006/2000A, CIO-based\n");
   printf ("=========================\n");

/*  CIP and CIO, IAU 2006/2000A. */

/* Report inputs to CIP calculation  */

   printf ("djmjd0  = %22.15f tt= %22.15f\n", djmjd0,TT);

   iauXys06a (djmjd0, TT, &x, &y, &s);

/*  Add CIP corrections. */
   x += dx06;
   y += dy06;

/*  Report CIP and s. */
   printf ("x = %22.15f\n",x/as2r);
   printf ("y = %22.15f\n",y/as2r);
   printf ("s = %22.15f\n",s/as2r);

/*  GCRS to CIRS matrix. */
   iauC2ixys (x, y, s, rc2i);

/*  Report.  */
   repmat ("NPB matrix, CIO-based (rc2i)", rc2i);

/*  Earth rotation angle. */

/*  Set tut to the value used in Tercel */

   printf ("djmjd0 = %f  date = %f  tut = %f\n",djmjd0, date, tut);

   era = iauEra00 (djmjd0, date+tut);
   printf ("era = %19.16f rad\n",era);
   printf ("era = %16.12f deg\n",era * 360.0/twopi);

   iauD2tf (9, era/twopi, &pm, IHMSF);
```

```
/*  Form celestial-terrestrial matrix (no polar motion yet). */
   iauCr (rc2i, rc2ti);
   iauRz (era, rc2ti);

/*  Report. */
   repmat ("celestial to terrestrial matrix (no polar motion)",rc2ti);
   iauCr (rc2ti,r3);

/*  Polar motion matrix (TIRS->ITRS, IERS 2003). */
   iauPom00 (xp, yp, iauSp00(djmjd0,TT), rpom);

/*  Form celestial-terrestrial matrix (including polar motion). */
   iauRxr (rpom, rc2ti, rc2it);

/* Transpose termat */
   transpose (termat, tertrans);
   transpose (sofmat, softrans);

/*  Report. */
   repmat ("celestial to terrestrial matrix", rc2it);
   repmat ("tercel_transposed matrix", tertrans);


/*  Copy for later comparison. */
   iauCr (rc2it, r4);

/*  Compare result to TERCEL_Transposed matrix */
   result1 = drot (r4, tertrans);
   printf ("%f\n", result1/ as2r);
   printf ("w/ pm result vs tercel = %20.10e uas\n", drot (r4, tertrans) *
           1.0e6 / as2r);

/* Compare SOFA w/P03 and SOFA w/o P03 */

   result2 = drot (r4, softrans);
   printf ("%f\n", result2/ as2r);
   printf ("sofa w/p03 vs w/o p03 = %20.10e uas\n", drot (r4, softrans) *
           1.0e6 / as2r);

}

/*----------------------------------------------------------------------*/

void repmat (char *s,double r[3][3])
{

   printf ("%s\n", s);
   printf ("%22.17f  %22.17f  %22.17f\n", r[0][0],r[0][1],r[0][2]);
   printf ("%22.17f  %22.17f  %22.17f\n", r[1][0],r[1][1],r[1][2]);
   printf ("%22.17f  %22.17f  %22.17f\n", r[2][0],r[2][1],r[2][2]);

   return;
}
```

```
/*----------------------------------------------------------------------*/

double drot (double rma[3][3],double rmb[3][3])
/*  Express the difference between two rotation matrices RMA,RMB as an
    amount of rotation R about some arbitrary axis.
*/
{
   double rmbt[3][3], rm[3][3], rv[3], r;

/*  Multiply the first matrix by the inverse of the second. */

   iauTr (rmb, rmbt);
   iauRxr (rmbt, rma, rm);

/*  Express the result as an r-vector. */

   iauRm2v (rm, rv);

/*  Return the magnitude (the amount of rotation in radians). */

   r = iauPm (rv);

   return r;
}

/*----------------------------------------------------------------------*/
void transpose (double r[3][3],double rt[3][3])
/*
  Transpose 3x3 matrix R ---> RT
*/
{
   short i, j;

   for (i = 0; i<3; i++)
   {
      for (j = 0; j<3; j++)
          rt[i][j] = r[j][i];
   }

}

/*______________________________________________________________________*/
void terceltest ()
{
/* Transform vectors from ITRS to GCRS */

   double tjdh, tjdl, xp, yp, delt = 65.25607389, vec1[3], vec2[3], tjd,
dx, dy,mobl,tobl,ee;
   short num = 0;
   FILE *In_Data = NULL;

   dx = +0.1750;
   dy = -0.2259;
```

```
/* Open the input file of Julian dates,CIO coords,ITRS vector */

   In_Data = fopen ("tercel-test-input.dat","r");

   while (!feof(In_Data))
   {
      fscanf
(In_Data,"%hi%lf%lf%lf%lf%lf%lf%lf",&num,&tjdh,&tjdl,&xp,&yp,&vec1[0],&vec
1[1],&vec1[2]);

/*  Set transformation method, accuracy level, and ut1-utc. */

      tjd = tjdh + tjdl;

/*      celpol (tjd,2,dx,dy);*/
      e_tilt (tjd,0, &mobl,&tobl,&ee,&dx,&dy);


/*  Rotate vec1 from ITRS to GCRS = vec2  */

      ter2cel (tjdh,tjdl,delt,1,0,0,xp,yp,vec1, vec2);
      printf ("%i    %20.17f    %20.17f
              %20.17f\n",num,vec2[0],vec2[1],vec2[2]);
   }

   fclose (In_Data);

}
```

Input Data File: tercel_matrix.txt

```
1     0.97310431770109063     0.23036382622411070    -0.00070316348296544
1    -0.23036380044102267     0.97310457063635536     0.00011854535854669
1     0.00071156016155651     0.00004662641201635     0.99999974575402495
```

Input Data File: sofa_matrix.txt

```
1    +0.97310431757363003    +0.23036382624733387    -0.00070333224579995
1    -0.23036379880468646    +0.97310457073561185    +0.00012088727913663
1    +0.00071226387930021    +0.00004438635469672    +0.99999974535497649
```

Input Data File: tercel-test-input.dat

```
1 2400000.5   54195.49999916581146 +0.0349282e0 +0.4833163e0 1.0  0.0  0.0
1 2400000.5   54195.49999916581146 +0.0349282e0 +0.4833163e0 0.0  1.0  0.0
1 2400000.5   54195.49999916581146 +0.0349282e0 +0.4833163e0 0.0  0.0  1.0
```

Output Data

```
termat
   0.97310431770109063    -0.23036380044102267     0.00071156016155651
   0.23036382622411070     0.97310457063635536     0.00004662641201635
  -0.00070316348296544     0.00011854535854669     0.99999974575402495
sofmat
   0.97310431757363003    -0.23036379880468646     0.00071226387930021
   0.23036382624733387     0.97310457073561185     0.00004438635469672
```

```
  -0.00070333224579995     0.00012088727913663     0.99999974535497649

date  2007 04  05
time    12 00    0.0 UTC

X_p =    0.034928200 arcsec
Y_p =    0.483316300 arcsec

ut1-utc =  -0.072073685000

dDpsi (1980) =  -0.055065500 arcsec
dDeps (1980) =  -0.006358000 arcsec

dx (2000) =   0.000172500 arcsec
dy (2000) =  -0.000265000 arcsec

dx (2006) =   0.000175000 arcsec
dy (2006) =  -0.000225900 arcsec

TT  = 2400000.5 +  54195.500754444445192
UT1  = 2400000.5 +  54195.499999165811460

=========================
IAU 2006/2000A, CIO-based
=========================

x =    146.915146447359746
y =      9.155115079288397
s =     -0.002200474981084\

NPB matrix, CIO-based (rc2i)
   0.99999974633944499    -0.00000000513882246    -0.00071226473007242
  -0.00000002647522726     0.99999999901497483    -0.00004438524282713
   0.00071226472959891     0.00004438525042571     0.99999974535441982

djmjd0 = 2400000.500000  date = 54195.000000  tut = 0.499999

era =  0.2324515536471452 rad
era =  13.318492965240 deg

celestial to terrestrial matrix (no polar motion)
   0.97310431757657001     0.23036382623316559    -0.00070333281884719
  -0.23036379878963870     0.97310457073901657     0.00012088854957536
   0.00071226472959891     0.00004438525042571     0.99999974535441982

celestial to terrestrial matrix
   0.97310431770097838     0.23036382622458479    -0.00070316348220013
  -0.23036380044149363     0.97310457063624356     0.00011854536661437
   0.00071156016266793     0.00004662640399541     0.99999974575402439

tercel_transposed matrix
   0.97310431770109063     0.23036382622411070    -0.00070316348296544
  -0.23036380044102267     0.97310457063635536     0.00011854535854669
   0.00071156016155651     0.00004662641201635     0.99999974575402495
```

```
w/ pm result vs tercel =     1.6738960753e+00 uas

sofa w/p03 vs w/o p03 =     4.8430687135e+05 uas
```